Implementing System Simulation of C^3 Systems Using Autonomous Objects

by
Ralph V. Rogers
Department of Industrial Engineering and Management System
University of Central Florida
P.O. Box 25000
Orlando, FL
32816-4450

ABSTRACT

The basis of all conflict recognition in simulation is a common frame of reference. Synchronous discrete-event simulation relies on the fixed points in time as the basic frame of reference. Asynchronous discrete-event simulation relies on fixed-points in the model space as the basic frame of reference. Neither approach provides sufficient support for autonomous objects. The use of a spatial template as a frame of reference is proposed to address these insufficiencies. The concept of a spatial template is defined and an implementation approach offered. Discussed are the uses of this approach to analyzes the integration of sensor data associated with C^3 systems.

INTRODUCTION

The analysis of Command, Control and Communication (C^3) systems requires an efficient, safe, and cost effective mechanism to investigate the effects of new technologies, control strategies, and tactics on system goals and performance. The mechanism of choice is computer simulation. Unfortunately, the fundamental modeling paradigms of current simulation modeling practices do not support integration of a wide range of phenomena of interest in a single model. As a result, multiple simulation models are used. Each such model relies on the modeling paradigm most compatible with the phenomena under consideration. Integrating the results from such multiple models requires artful and experienced analysts.

The major shortcoming of this multiple modeling approach is the inability to focus on the interaction of many differing phenomena. For example, consider the following situation of interest. New satellite navigation systems have become available for aircraft use. Their accuracy and precision are well established. Several airlines and some private aircraft are equipped with the new system. Additionally, current VOR/DME systems are still in effect and used on all IFR aircraft even those with the new satellite navigation system. The general accuracy and precision of these systems are also well known as well as their areas of coverage. Other sensors such as radar and secondary radar (i.e. transponder) are available to the air traffic controller. These sensors' areas of coverage and accuracy are also known. Additionally, the pilot of each aircraft can report her position *as they understand it* to the controller. The controller, likewise, can report the location of the aircraft to the pilot as the *controller understands it*. The question of interest is: given a percentage of both commercial and private aircraft with the satellite navigation system, what is the effect on overall separations between aircraft and between aircraft and ground objects? Associated with this question is the establishment of the rules/procedures for *information fusion* from all the possible sources of information for both pilot and controller. This is especially of concern because of non-overlapping blind areas among the various sensors and navigation systems. In effect, the information fusion rule/procedures become the independent variables of this model along with the percentage mix of satellite navigation equipped and non-equipped aircraft.

Implementing this scenario with the existing simulation paradigms of continuous, synchronous, and asynchronous modeling would be, at best, very difficult. The current paradigms simply do not lend themselves to modeling such a situation. Continuous simulation modeling could not support the complex of objects and spatial environment of this scenario. Synchronous modeling would be overwhelmed by the extensive overhead necessary to create and execute the model. Asynchronous modeling could not effectively or efficiently capture the strong spatial performance measures associated with this scenario. Moreover, each paradigm faces other lesser problems in implementing the situation presented. What is required is a simulation modeling paradigm which is supportive of the situations described above. The simulation modeling paradigm offered for this purpose is object-oriented modeling with autonomous objects.

AUTONOMOUS OBJECTS

Autonomous objects posses full freedom of movement and decision making within a simulation model. Autonomous objects may alter their temporal and spatial goal trajectories conditional upon their current state and the state of the rest of the system. Autonomous Object positional accuracy can be arbitrarily precise within a model. Autonomous objects are obviously desireable in modeling and simulating C^3 systems. Such objects

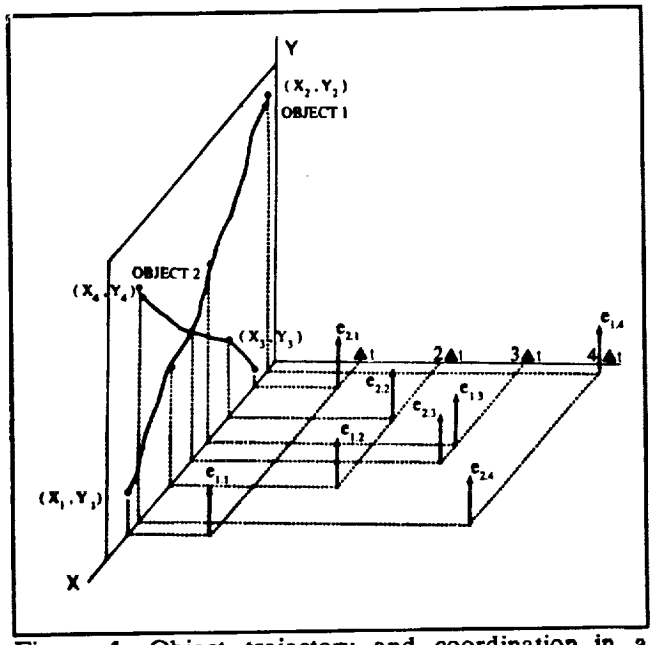

Figure 1. Object trajectory and coordination in a synchronous DES.

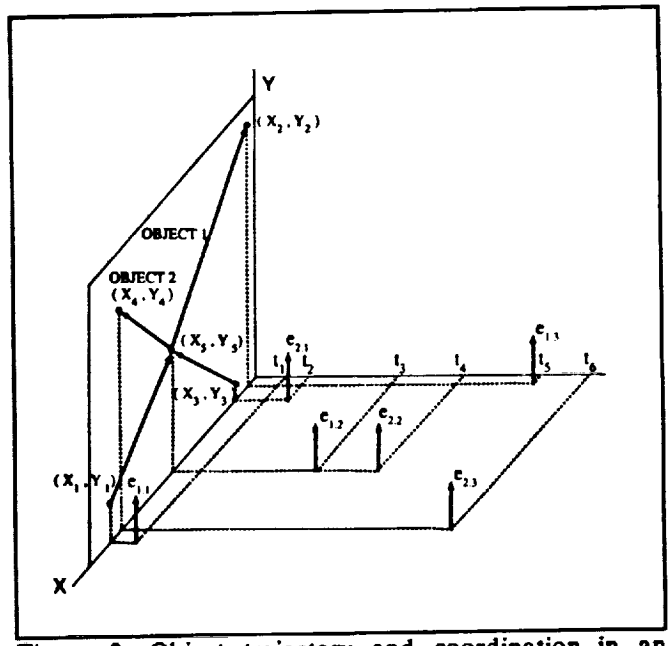

Figure 2. Object trajectory and coordination in an asynchronous DEVS model.

objects more closely approximate the highly independent and dynamic behavior associated with the constitutes of a C^3 environment such as aircraft and surface ships. However, implementing autonomous objects into C^3 simulation models is typically achievable only through limited hybridization of two competing simulation modeling paradigms--continuous simulation and discrete-event simulation (DES).

Autonomous objects are generally associated with continuous simulation models of relatively narrow environments composed of small numbers of autonomous objects. Continuous model are typically composed of a number of differential equations which are solved numerically. However, in dense, highly dynamic, complex environments involving multiple autonomous objects such as C^3 systems, the modeling and computational resource demands of such models generally limit their use. The DES approach offers a more computationally efficient approach. However, current approaches for incorporating autonomous objects into DES models relies on integrating continuous simulation approaches into the DES environment. While this hybridization provides some added capabilities, it also is computationally limited in the number of autonomous objects it can efficiently manage.

The key challenge hindering further use of autonomous objects in DES based C^3 simulations surrounds establishing common reference frames and referencing methodologies for object decision making (i.e. object coordination). This paper discusses the concept of object coordination and offers an approach to coordination which increases the utility of autonomous objects in DES C^3 models.

AUTONOMOUS OBJECTS IN DES

The two established methods of DES modeling are synchronous and asynchronous modeling. In synchronous simulation modeling, time is advanced in fixed increments. The model is examined only at regular intervals defined by the time increment used. At every advance of the clock, the state of each object, entity and process in the simulation must be updated. Conflicts and resolution must be identified and actions implemented. Conditional operational decisions are made (and coordinated) at these fixed time intervals. The reference for coordination of object actions is the common time defined by the increment. Synchronous modeling is an advantageous approach when it is desired for a certain event to occur when a particular condition is satisfied (or identified). Figure 1 provides a graphical illustration of the synchronous approach for two autonomous objects moving through a plane. Notice that each object must establish its relationship to the other object at each clock increment Δt. Synchronous modeling has three major disadvantages. First, such models are hard to implement in software. Second, nonsimultaneous events may be treated as simultaneous causing priority and sequencing problems. Third, to obtain sufficiently accurate performance measures, it may be necessary to make Δt very small. As Δt decreases, the number of times the model must be updated (and objects synchronized) increases. This increased updating also increases the computational resources and time required to exercise the model [1].

In asynchronous simulation modeling, the next scheduled event (i.e. state change) defines the next increment of time that advances the clock. Event can occur at anytime. Thus, objects

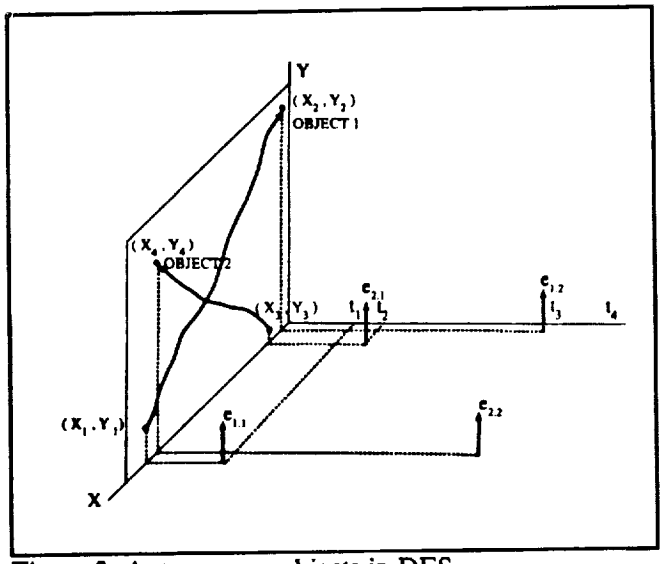

Figure 3. Autonomous objects in DES.

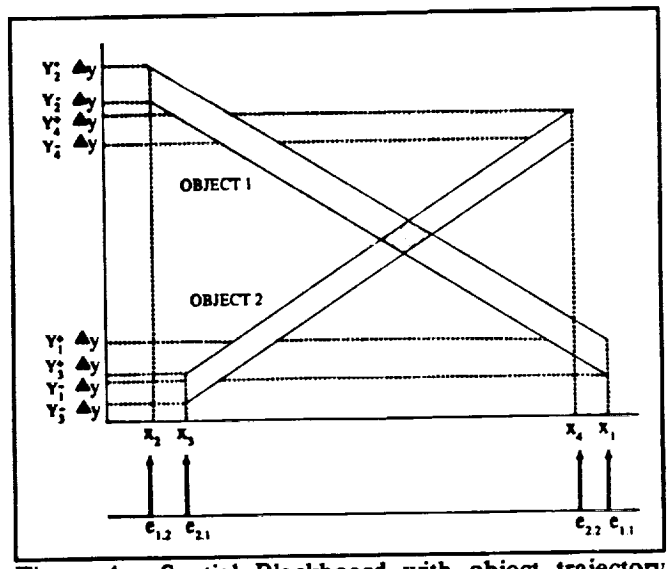

Figure 4. Spatial Blackboard with object trajectory polygons.

may have their states updated at different times and the resulting increments between time advances may vary widely through the course of a simulation exercise. However, a common reference is still required to identify and resolve conflicts and other interactions between objects (i.e. coordination). Such coordination of object interactions and dependencies in an asynchronous modeled system requires operational decision points. These decision points are not fixed in time as in synchronous simulation. Instead, they are fixed in model space. That is, the asynchronous solution to model object management is to fix the decision points in model space thereby fixing the spatial increments of the model. Conditional decision are normally triggered by the arrival of objects or entities to some point in the model space. Conditions referenced to simulated time may be difficult or impossible to implement correctly in an asynchronous model because the model may not recognize that the condition is true until the next event occurs. By that time, the condition may be false again [1].

The underlying paradigm of asynchronous modeling is node and arc networks. Nodes represent decision points and arcs represent specific distances and/or time. Conflicts are resolved at the nodes. Thus, the frame of reference for asynchronous modeling is the fixed network nodes or similar simulation construct corresponding to fixed points in model space.

Figure 2 provides a graphical illustration of the corresponding asynchronous simulation of two objects moving on a plane. Note that the trajectory paths are explicitly defined as part of the model. The coordination point for the two objects is point X_5Y_5 of X-Y plane. In this case, the decision or control logic must only consider the objects and interactions associated with and prior to events $e_{1,2}$ and $e_{2,2}$.

Figure 2 also illustrates the difficulty confronting autonomous object implementation in asynchronous simulation. Autonomous objects by definition must have the freedom to modify their trajectory through model space, to go where their goals and operational rules specify. However, to coordinate the actions of these or any objects, some common reference frame must be established. Figure 3 illustrates this problem for autonomous objects in an event driven simulation. In this example the objects schedule their next event (i.e. an arrival) some time in the future (t_3 and t_4). However, with no common points defined for them in the model in either time or space, there is no mechanism for coordinating object decision making.

What is desired is to allow each autonomous object to schedule its next event anywhere in the model space its operational rules permit, determine if there are any conflicts with any other objects in the model in achieving that next event, and implement conflict resolutions strategies among objects when necessary. How objects recognize and resolve conflicts are the basic issues of to be addressed in implementing autonomous objects in DES models. General strategies for these resolving these issues are discussed in the following.

CONFLICT IDENTIFICATION STRATEGIES

A simple strategy requires each object to schedule its next model space position (and associated event) according to its own objectives. The object would then poll ever other object in the model to determine if the newly scheduled event would precipitate any conflicts. Conflicts would be resolved based upon the model's conflict resolution procedures. This is similar to the general approach used in synchronous modeling.

Unfortunately, such a simple conceptual strategy suffers from the same basic demand for computational resources as the synchronous modeling. Moreover, computational resource demand grows with at least the square of the number of objects

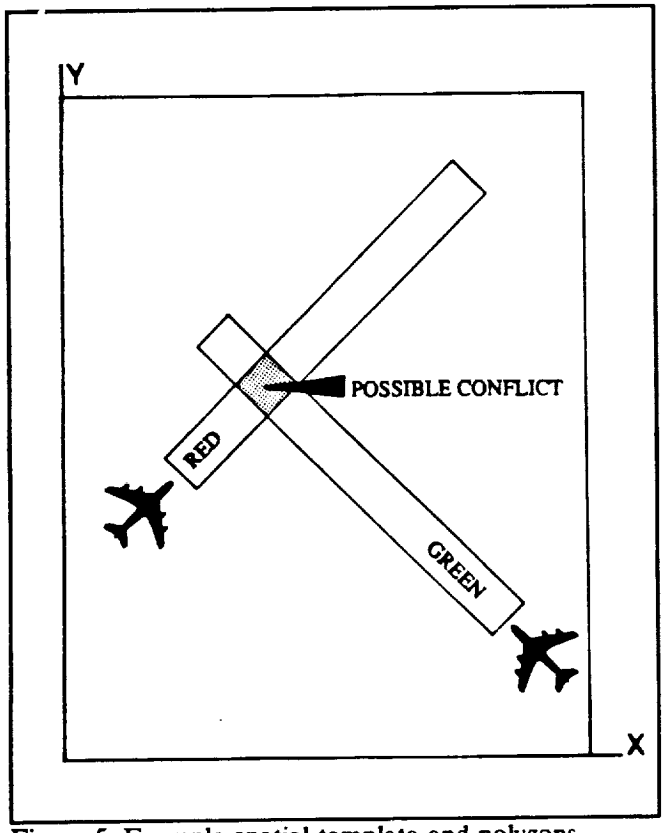

Figure 5. Example spatial template and polygons.

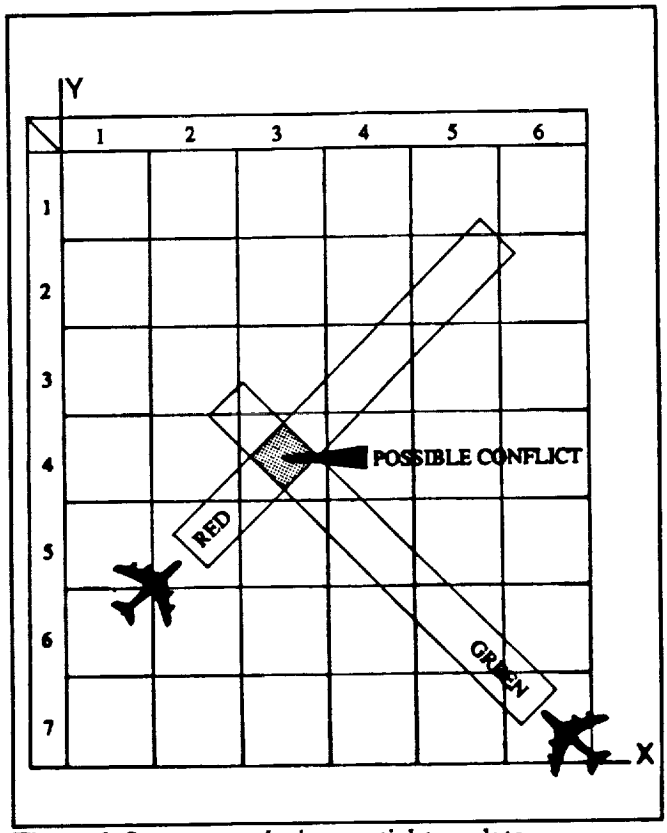

Figure 6. Sectors overlaying spatial template.

in the model. What is needed is an approach which enables conflict recognition between objects while minimizing the number of objects and number of communication channels and interchanges which must be maintained between objects.

The conflict identification strategy pursued in the author's research uses the concept of a spatial template. Under this concept, all objects in the model space are represented as geometric shapes. To simplify, all object model shapes are polygons. Dynamic objects (e.g. airplanes, ships, guided vehicles, weather, etc.) are also represented by polygons but polygon size and shape is based on the objects model space trajectory. The parameters of the associated polygon of the dynamic objects in the spatial template are defined by the originating object event and the next scheduled event for that object as well as unique characteristics of the object.

For example, the trajectories of the two objects of the previous discussions could be represented as two polygons in the X-Y plane of their movement. The origin of Object 1's trajectory is point X_1,Y_1 and the end of its trajectory is point X_2,Y_2. These points correspond to events $e_{1,1}$ and $e_{1,2}$. Similarly for Object 2, its origin and end points are X_3,Y_3 and X_4,Y_4, respectively. The corresponding events are $e_{2,1}$ and $e_{2,2}$. The polygons on the spatial template for each of these objects could be defined by the origin and end points and by a Δy for each object. Thus, a polygon representative of Object 1 would be defined by the four points $(X_1,Y_1+\Delta y)$, $(X_1,Y_1-\Delta y)$, $(X_2,Y_2+\Delta y)$, and $(X_2,Y_2-\Delta y)$. A polygon representation of Object 2 would be defined by the four points $(X_3,Y_3+\Delta y)$, $(X_3,Y_3-\Delta y)$, $(X_4,Y_4+\Delta y)$, and $(X_4,Y_4-\Delta y)$. The two corresponding polygons and spatial template are illustrated in Figure 4.

This static representation in two dimensions implies a third dimension (time) by the extent of the polygon from the point of the arrival event. This static representation may also be regarded as a most probable model space trajectory for the object. Uncertainty in the proposed model space trajectory may be reflected in the shape and extent of the object polygon (e.g. the Δy's). Figure 4 illustrates this concept. ***Potential*** conflicts are identified when the representational polygons of objects intersect. Once intersection is detected, objects may then resolve conflicts. Clearly, the key methodological issues are associated with how to efficiently identify polygon intersections in the spatial template.

The problem of how to determine if any two geometric objects intersect in a given coordinate system is a well known and widely addressed problem. Efficient and easily implementable methodologies are readily available. The problem with polygon intersection detection for the spatial template approach is the determination of what is an efficient way to identify which objects should be tested to determine if they intersect? Testing between all objects in the system every time a new event is

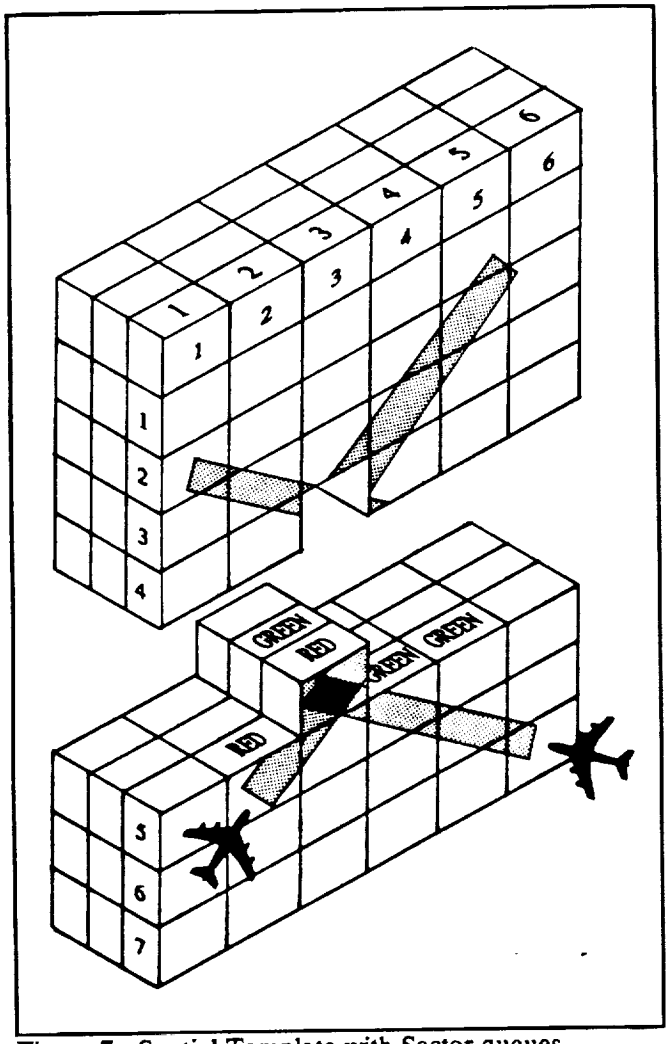

Figure 7. Spatial Template with Sector queues.

scheduled merely recasts the problems of synchronous modeling.

The spatial template is fundamentally concerned with the identification of the ***Potential*** conflicts. That is, to identify those objects whose model space trajectories may compete for the same model space resources (e.g. space) at the same time. The goal is to reduce the message traffic between model objects necessary to determine if conflicts will occur. The approach taken is to graphically represent the model space trajectory of model system objects and to identify the intersection of trajectories before allowing the object to proceed. How to identify these potential conflicts in an efficient manner is dependent on the how to represent the model space information and the object trajectory information associated with trajectory polygons and scheduled simulation events.

The proposed spatial template approach to representing this information is to partition a two or three dimensional cartesian model space into equal sectors. The model space trajectory of an object represented by its associated polygon is defined within the model space cartesian coordinate system (See Figure 5) as discussed before. The Sector partitioning then overlays the model space cartesian system (See Figure 6). The Sectors are identified by a coordinate sector numbers (e.g. sector 1,3). The Sectors through which the trajectory polygon overlays and/or intersects are identified. Associated with each sector then is a list of objects whose trajectory is schedule to go through some part or all of that sector (See Figure 7). When new object trajectories are added to a the sector list, a check is made to determine if any other object trajectories are also associated with that Sector. If an object trajectory is already associated with a Sector, then a potential conflict exists. When potential conflicts are identified, the model communicates with each object in the identified Sector to determine if an actual conflict exists. When conflicts do exist, the model may then employ its conflict resolution strategies to remove the conflict.

The data structure employed to maintain the list of objects associated with spatial template Sectors is a dynamic array of queues. The Sectors of the model space become elements in the array. Each element in the array is a queue. As object trajectories cross a sector, the name of the object is added to the sector (i.e. queue). Once a new model trajectory has been established for an object, the name of the object is remove from the sectors associated with the old object trajectory.

In the object-oriented software implementation, each sector element is defined as a queue object. The use of objects to describe sectors enables the exploitation of a dynamic array. A dynamic array will only use the memory necessary for the active sector objects. Thus, if no sectors have objects associated with them, they are not created, and do not require computational resources. Likewise, as a sector becomes empty, that sector object may be disposed of, freeing computational resources.

Implementing the spatial template as described above in a DES provides an approach which supports autonomous object modeling. The spatial template approach reduces the coordination overhead required for autonomous objects in both synchronous and asynchronous simulation. Practical programming considerations dictate an object oriented approach is required for the software implementation.

C³ MODELING IMPLICATIONS OF AUTONOMOUS OBJECTS

The use of autonomous objects and the object oriented modeling paradigm in general would allow the capture of the scenario described earlier. Sensors could be modeled as just another unique object in the model. A radar, for example, would define its pattern search with a polygon on the spatial template (Figure 8). Aircraft identified within the polygon pattern would be asked to report their true position and altitude in the model space. The sensor would then correct for blind areas and altitude. Uncertainty, imprecision, and other anomalies of the radar performance could be used to modify the position information before forwarding it on to a controller. Similarly, information from navigation objects and controller objects could be provided to a pilot object which may direct the

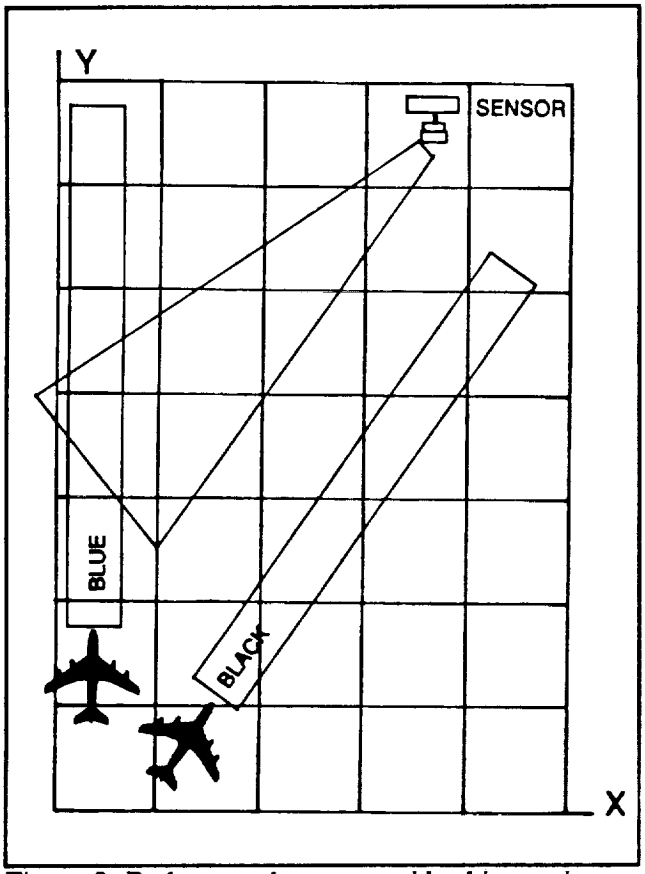

Figure 8. Radar search pattern with object trajectory polygons in spatial template.

aircraft object.

Ground obstructions such as buildings, towers, and mountains could be represented on the spatial template with polygons. Control strategies by pilots or controllers would respond accordingly when such ground obstruction were detected. Of course, flawed strategies could lead to collisions with ground obstructions or other aircraft.

Building simulation models with autonomous objects in an overall object-oriented modeling scheme frees the simulationist from the traditional network paradigm of large system modeling. In the autonomous object approach, the corresponding phenomena of interest can be identified with its corresponding "real-world" manifestation (i.e. object). Modeling autonomous object includes including the behaviors of interest with the object model. In traditional DES modeling, the model emphasis is on external processes which operate ON and implement changes TO objects of interest. The objects of interest from the simulation model perspective are little more than tokens with slots being passed from one location or process to another. The modeler is required to impose a interpretation of the structure of the interactions of the model objects when constructing the model.

Many times in system engineering one is interested in the overall effect which arises when the behaviors of individual objects or entities are brought together. The goal is to observe the structural effects on the system (or the system defined) by individual objects pursuing their own goals as dictated by their own behaviors and goals. In traditional system/process modeling the system structure is assumed in the modeling process. Only the questions of system performance are left to be resolved. Object modeling with autonomous objects permits the investigation of the effects on system structure of new behavior and rules and not just on specific performance measure of a system operating under assumptions the new behaviors may undermine.

SUMMARY

Clearly, the use of the object-oriented modeling paradigm provides a more natural and intuitive modeling approach for C^3 systems. Autonomous objects in object models provide the capability to capture the wide variety of phenomena necessary to study system interactions in large scale, highly dynamic environments. Perhaps just as important, autonomous objects enable the management of technical, site specific data and operational RULES within a model. Unfortunately, the decision making frame of references in traditional synchronous and asynchronous DES modeling severely limit the ability of these approaches to support autonomous object models.

The spatial template offers a frame of reference which addresses the limitations of the traditional DES models. Thus, the spatial template basis of model coordination and the use of object-oriented software provides a mechanisms for implementing autonomous objects in a DES modeling paradigm and computer software. The availability of such simulation tools has the potential to change the scope and level of discrete event simulation use in systems analysis, in general, and C^3 systems analysis, in particular.

REFERENCES

[1] P. Bratley, B.L. Fox, and L.E. Schrage. *A Guide to Simulation*, 2nd ed. New York: Springer-Verlag, 1987.

ACKNOWLEDGEMENTS

This work was supported in part by the Federal Aviation Administration and the National Aeronautics and Space Administration under NASA-AMES Grant NAG 2-625.